Les requêtes

pour Overpass Turbo

Table des matières

Introduction

OpenStreetMap, est une base de données géographique, mondiale, libre, gratuite et collaborative. Les données de cette base sont donc libres d'accès sous condition d'accepter la licence ODbL.

Cette licence vous permet d'exploiter les données OpenStreetMap commercialement ou non, à condition de garder la même licence sur les modifications apportées et de mentionner explicitement la source et les contributeurs.

Les données OpenStreetMap sont disponibles depuis plusieurs serveurs, à partir de différentes applications ou site Internet.

Ce document montre une des manières permettant de récupérer les données, en utilisant les requêtes Overpass Turbo. Ces requêtes sont écrites soit en Extensible Markup Language (XML), soit en Query Language (QL). Nous allons plutôt mettre l'accent sur le langage QL qui permet d'aller plus loin dans les détails de requête. Mais nous garderons le langage XML pour pouvoir faire la différence entre ces 2 langages.

Pour faciliter la navigation dans ce document, les requêtes seront bordées à gauche d'une ligne :

- Continue pour les requêtes en langage ql
- Pointillée pour les requêtes en langage xml

Les données OSM peuvent être récupérées de 3 manières différentes :

- Directement depuis Internet sur http://overpass-turbo.eu/
- Depuis l'extension QuickOSM de QGIS (Extension>Installer/gérer les extensions)
- A partir de JOSM (Fichier>Téléchargement depuis API ou Alt+Maj+Bas)

Les requêtes

Comment ça marche

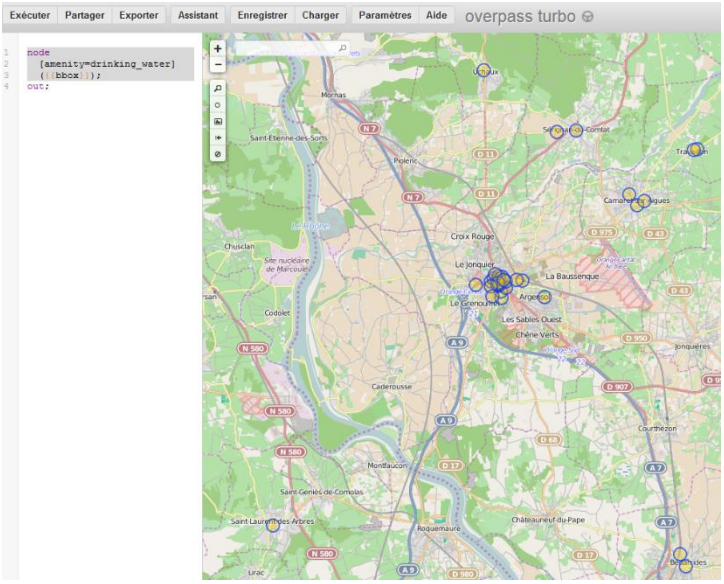
Pour récupérer les données OpenStreetMap, il faut envoyer une requête URL à un serveur distant qui renvoi un fichier xml en réponse.
Pour lancer cette requête, nous avons besoin des éléments suivants :
Serveur[option1] [option2][…] ; (type d'objet + clé/valeur + emprise géographique ;) ;métadonnées ;

En passant par un site comme http://overpass-turbo.eu/ , il est possible de construire sa requête via un assistant.

**** Requête XML ****

Nous allons essayer de décortiquer une requête XML réalisée dans overpass-turbo. Pour prendre un exemple, nous allons récupérer les points d'eau potables situés dans l'emprise de la carte ci-contre. Cette requête simple est la suivante :

```
node
 [amenity=drinking_water]
  ({{bbox}});
out;
```



Signification de la requête
node *= on récupère des objets points*
[amenity=drinking_water] *= on récupère les objets qui ont comme clé « amenity » et comme valeur pour cette clé « drinking_water »*
({{bbox}}); *= on récupère les données qui se trouvent dans l'emprise de la carte à droite*
out; *= fin de la requête*

**** Requête QL ****

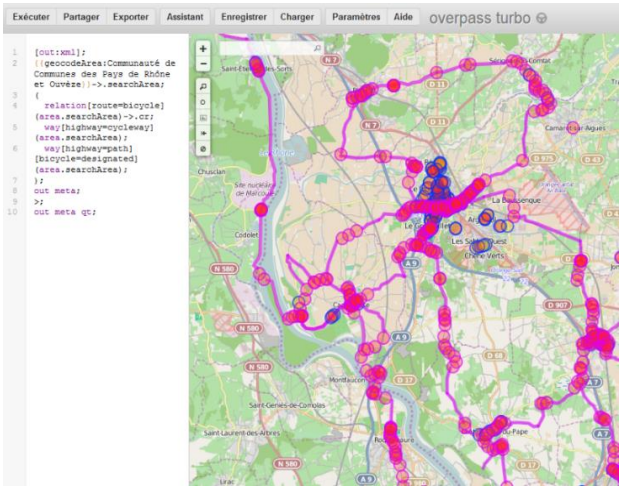
En passant par l'assistant d'overpass-turbo, on obtient une requête en QL. Par exemple, si on écrit *route=bicycle in 'Communauté de Communes des Pays de Rhône et Ouvèze'*, nous obtenons la requête suivante :

```
[out:json][timeout:25];
{{geocodeArea:Communauté de Communes des Pays de Rhône et Ouvèze}}->.searchArea;
 (
  node["route"="bicycle"](area.searchArea);
  way["route"="bicycle"](area.searchArea);
  relation["route"="bicycle"](area.searchArea);
);
out body;
>;
out skel qt;
```

Les requêtes pour Overpass Turbo

Nous allons modifier cette requête QL afin de récupérer les réseaux de vélo-tourisme d'une intercommunalité, ainsi que les pistes et les voies cyclables. Nous souhaitons aussi avoir une sortie xml. Voici la requête :

```
[out:xml][timeout:25];
{{geocodeArea:Communauté de Communes des Pays de Rhône et Ouvèze}}->.SA;
(
  relation[“route”=”bicycle”](area.SA)->.cr;
 way[“highway”=”cycleway”](area.SA);
way[“highway”=”path”][“bicycle”=”designated”](area.SA);
);
out meta;
>;
out meta qt;
```



Signification de la requête

[out:xml]; *= on récupère un fichier en xml.*

[*timeout:25*]; *= définit le temps maximum d'exécution de la requête en seconde.*

{{geocodeArea(…)}}->.SA; = on récupère les données qui se trouvent dans l'emprise de la Communauté de Communes des Pays de Rhône et Ouvèze. L'orthographe de l'emprise doit être exactement la même que dans OpenStreetMap (casse, tiret,…). Cette emprise est envoyée dans une variable (->.SA)

[“highway“=“cycleway“] *= C'est le Prédicat. On récupère les objets qui ont comme clé « highway » et comme valeur de cette clé « cycleway »*

(

relation[“route“=“bicycle“](area.SA)->.cr *= on récupère les relations qui ont comme clé « route » et comme valeur de cette clé « bicycle »*

way[“highway“=“cycleway“](area.SA); *= on récupère les chemins qui ont comme clé « highway » et comme valeur de cette clé « cycleway»*

way[“highway“=“path“][“bicycle“=“designated“](area.SA); *= ET logique, on récupère les chemins qui ont comme clé « highway» et comme valeur de cette clé « path » ET qui ont comme clé « bicycle» et comme valeur de cette clé « designated»*

); *= OU logique. Les parenthèses sont des unions. Ici, on récupère les relations route=bicycle OU les way highway=cycleway OU les way highway=path ET bicycle=designated.*

out meta; *= on récupère les objets (ways) et les métadonnées des objets. Voir ci-après pour le détail*

>; *= on récupère tous les objets dépendants (ici, les nodes des ways)*

out meta qt; *= on récupère aussi les métadonnées des objets dépendants. Voir ci-après pour le détail*

Cette requête est plus complexe mais a l'avantage d'être plus précise dans les objets attendus en retour. Nous allons détailler les paramètres et les options qu'il est possible d'ajouter à la requête QL.

Prédicat

Définition

Le prédicat est la partie de la requête qui définit la propriété des objets à récupérer (["amenity"="hospital"], ["higway"="residential"],…).
Il est défini de la manière suivante : ["clé" <opérateur> "valeur"].
["amenity"="hospital"] renvoi les objets qui contiennent la clé "amenity" et dont la valeur pour cette clé est "hospital".
En mettant 2 prédicats l'un derrière l'autre, cela équivaut à un « ET logique ». Par exemple, ["highway"="path"]["bicycle"="designated"] retourne les objets highway=path ET bicycle=designated.

Opérateurs

["clé" = "valeur"] : retourne les objets pour lesquels la clé est strictement égal à la valeur. Ex ["amenity"="hospital"]
["clé" ~ "valeur"] : retourne les objets pour lesquels la clé contient la valeur. Ex ["name"~"Clermont"]
["clé" ~ "^valeur"] : retourne les objets pour lesquels la clé commence par la valeur. Ex ["name"~"^Jac"]
["clé" ~ "valeur$"] : retourne les objets pour lesquels la clé fini par la valeur. Ex ["name"~"ville$"]
["clé"] : retourne tous les objets contenant la clé, quelle qu'en soit la valeur. Ex ["highway"]
["clé" ~ "."] même chose que ci-dessus car en sql, le « . » équivaut à « * ». Ex ["highway"~"."]
["clé" !~ "valeur"] : retourne les objets pour lesquels la clé ne contient pas la valeur. Ex ["place" !~"city"]
["clé" !~ "."] : retourne les objets pour lesquels la clé est manquante. Ex ["wheelchair"!~"."]

Paramètres de la requête

Il y a 3 types de paramètres pouvant être définis dans la requête : la durée de la requête, le nombre d'objets téléchargés et le format de sortie.

**** Durée de la requête ****

Le paramètre [timeout:25] définit le temps maximum autorisé pour d'exécution de la requête en seconde (180s par défaut). Au-delà de cette durée, le serveur annulera la requête. Si le temps mis dans le timeout est trop long, le serveur rejettera automatiquement la requête avant même son exécution.

**** Nombre d'objets téléchargés ****

Il est aussi possible de limiter l'utilisation de la mémoire RAM du serveur en ajoutant [maxsize:1073741824]. Le nombre correspond à un entier non négatif exprimé en bytes. La valeur par défaut est 536870912 (512 MB). Il a les mêmes effets d'annulation ou de rejet de la requête de la part du serveur que pour le timeout.

**** Format de sortie ****

Le paramètre [out:] défini le format de sortie du fichier de données. Il ne doit pas être confondu avec l'action « out » de fin de requête. Il existe 5 types de formats :

- [out:xml] pour récupérer un fichier xml
- [out:json] pour récupérer du json en sortie (/!\, ce n'est pas du geojson)
- [out:csv] pour récupérer un fichier csv. Il a besoin de paramètres supplémentaires pour définir les champs, les en-têtes et les séparateurs.
- [out:custom] pour récupérer un fichier au format personnalisé
- [out:popup] pour récupérer un fichier au format popup

Options des objets

Le paramètre « out » en fin de requête permet de filtrer les informations que l'on souhaite récupérer. Ce paramètre peut s'accorder avec :

- ids: pour récupérer seulement les identifiants des objets
- skel: pour récupérer seulement les informations nécessaires à la géométrie des objets (coordonnées des nœuds et des chemins, identifiants des membres des chemins et des relations)
- body: pour récupérer toutes les informations nécessaires pour utiliser les objets. Par exemple, les clés de tous les objets et les rôles des membres de chaque relation
- tags: pour récupérer seulement les identifiants et les clés de chaque objet (sans coordonnées ni membres des relations)
- meta: pour récupérer tout ce qui est connu de chaque objet, y compris les versions, les changeset, timestamp, user data de la dernière modification de l'objet.

Il est aussi possible d'ajouter des informations dérivées :

Les requêtes pour Overpass Turbo

- bb: pour ajouter les emprises de chaque objet. Pour les nœuds, c'est l'équivalent de "geom". Pour les chemins, c'est l'emprise de tous les nœuds. Pour les relations, c'est l'emprise de tous les nœuds et les chemins membres, les relations membres ne sont pas inclues.
- center: pour ajouter les coordonnées du centre de l'emprise définie ci-dessus pour les chemins et les relations. Note: Le centre peut ne pas être à l'intérieur du polygone.
- geom: pour ajouter la géométrie complète de chaque objet. Cela ajoute les coordonnées de chaque nœud, y compris ceux appartenant à un chemin ou à une relation et cela ajoute la séquence des membres « nd » avec leurs coordonnées à toutes les relations. L'attribut « geom » peut être suivi d'une « bounding box » au format "(Sud, Ouest, Nord, Est)". Dans ce cas, seules les coordonnées à l'intérieur de cette emprise seront récupérées. Pour les chemins, la première coordonnée à l'extérieur de l'emprise sera récupérée afin de former les segments proprement.

Il est aussi possible de trier les objets retournés par la requête. Par défaut, c'est l'attribut asc qui est utilisé.

- asc: Tri selon l'id
- qt: Tri selon l'index « QuadTile » correspondant à peu près à un tri géographique et significativement plus rapide qu'un tri par id
- Un nombre entier non négatif correspondant au nombre maximum d'objets à récupérer (aucune limite par défaut)

Enfin, l'option « ._ » permet de récupérer les nœuds des chemins et les nœuds et chemins des relations.

- (node(43.98,4.70,44.27,4.99);way(43.98,4.70,44.27,4.99);relation(43.98,4.70,44.27,4.99););(._;>;);out meta;

Emprises géographique et étendues

De manière générale, l'emprise géographique s'écrit de la manière suivante : (south,west,north,east), exemple (43.98,4.70,44.19,4.96)
Dans Overpass turbo, pour récupérer les données se trouvant à l'intérieur d'une entité nommée, on peut aussi utiliser le paramètre {{geocodeArea:« » }}->.SA, en mettant entre « » le nom de l'entité, par exemple, {{geocodeArea:«Bédarrides»}}. Cette emprise est envoyée dans la variable « .SA » qui sera utilisée dans les prédicats de la requête de la manière suivante : way[highway=cycleway](area.SA);

Overpass turbo utilise ces requêtes Overpass "étendues" dans plusieurs cas afin de fournir des raccourcis pratiques pour des morceaux de code fréquemment utilisés comme la {{bbox}} de la carte. Ces extensions permettent également à des données additionnelles et/ou des réglages d'être utilisés pour chaque requête, comme un {{style}}MapCSS (feuille de style) par exemple.

De telles extensions sont toujours ajoutées à la requête Overpass habituelle par un modèle *mustache* qui commence avec deux accolades ouvertes {{ et finit avec deux accolades fermées}}. Les emprises possibles sont :

- {{bbox}} : Il s'agit de la boîte englobante de la fenêtre de la carte ouverte.
- {{center}} : Il s'agit des coordonnées du centre de la fenêtre de la carte ouverte.
- {{date:string}} :
- {{geocodeId:name}} :
- {{geocodeArea:name}} :
- {{geocodeBbox:name}} :
- {{geocodeCoords:name}} :

Il est aussi possible de récupérer les objets situés 500m autour d'un autre objet. Pour cela, nous utilisons *node["highway"="bus_stop"](around:500,{{geocodeCoords:Groupe ESC Clermont}});*

Date des données

Pour récupérer des données à une date spécifique, il faut ajouter une nouvelle ligne après le timeout et mettant [diff:"2012-09-14T15:00:00Z"] (pour des données datant du 14/09/2012 à 15:00).
Pour récupérer des données entre 2 dates, il faut mettre [diff:"2012-09-14T15:00:00Z","2012-09-21T15:00:00Z"].
[out:xml][timeout:25];

Les requêtes pour Overpass Turbo
[diff:"2012-09-14T15:00:00Z","2012-09-21T15:00:00Z"] ;
(

(change:"date1","date2")

Autres astuces

- La syntaxte *out; >; out;* peut aussi s'écrire *(._; >;);* car «._ » est le résultat du prédicat pour lequel on récupère les objets enfants (nodes des ways ; nodes et ways des relations).
- La syntaxe *rel(385735); node(r); out; >; out;* nous permet de récupérer tous les nœuds de la relation 385735, quels que soient leurs rôles « *(r)* ».
- On peut récupérer ces nœuds en fonction de leurs rôle dans la relation (ex : *node(r:"forward_stop");*)
- On peut aussi filtrer sur les nœuds selon le prédicat *node(r)["shelter"~"yes"];*
- Cela peut aussi se faire en 2 étapes :
    - *node(r:"forward_stop")->.bs;*
    - *node.bs["shelter"="yes"];*
- ou encore :
    - *node(r:"forward_stop");*
    - *node._["shelter"="yes"];*
- Pour cette relation, on peut récupérer les ways qu'elle contient avec *rel(385735); way(r);*

Conclusion

Nous avons vu les principales règles nous permettant de réaliser des requêtes Overpass. Nous allons voir maintenant une série d'exemples pour nous familiariser à la construction de nos requêtes.

Quelques exemples de requêtes

Utilisation des attributs de métadonnées

Nous allons balayer les différentes options permettant de récupérer les attributs et les métadonnées. En vert, nous pouvons voir les données qui ont été ajoutées en plus par rapport à la requête précédente.

**** Option ids ****

Récupération des identifiants des objets

La requête suivante :

```
[out:xml][timeout:25];
(
  node["amenity"]({{bbox}});
  way["amenity"]({{bbox}});
  relation["amenity"]({{bbox}});
);
out ids;
```

renvoi les données suivantes (extrait) :

```
<?xml version="1.0" encoding="UTF-8"?>
<osm version="0.6" generator="Overpass API">
<note>The data included in this document is from
www.openstreetmap.org. The data is made available
under ODbL.</note>
<meta osm_base="2015-12-29T10:08:02Z"/>
  <node id="860901889"/>
  <node id="861724615"/>
  <way id="327755238"/>
  <way id="374226459"/>
  <relation id="1654142"/>
  <relation id="3371895"/>
</osm>
```

**** Option skel ****

Récupération des identifiants et de la géométrie des objets

La requête suivante :

```
[out:xml][timeout:25];
(
  node["amenity"]({{bbox}});
  way["amenity"]({{bbox}});
  relation["amenity"]({{bbox}});
);
out skel;
```

renvoi les données suivantes (extrait) :

```
<?xml version="1.0" encoding="UTF-8"?>
<osm version="0.6" generator="Overpass API">
<note>The data included in this document is from
www.openstreetmap.org. The data is made available
under ODbL.</note>
<meta osm_base="2015-12-29T10:13:01Z"/>

  <node id="860901889" lat="44.1377449"
lon="4.8093706"/>
  <node id="861724615" lat="44.1352485"
lon="4.8121347"/>
  <way id="327755238">
    <nd ref="3345693509"/>
    <nd ref="3345693515"/>
  </way>
  <way id="374226459">
    <nd ref="3775137769"/>
    <nd ref="3775137762"/>
  </way>
  <relation id="1654142">
    <member type="way" ref="43671610"
role="label"/>
    <member type="way" ref="43671611"
role="label"/>
  </relation>
  <relation id="3371895">
    <member type="way" ref="251208243"
role="outer"/>
    <member type="way" ref="134904848"
role="inner"/>
  </relation>

</osm>
```

**** Option body ****

Récupération des identifiants, de la géométrie et des attributs des objets

La requête suivante :

```
[out:xml][timeout:25];
(
  node["amenity"]({{bbox}});
  way["amenity"]({{bbox}});
  relation["amenity"]({{bbox}});
);
out body;
```

renvoi les données suivantes (extrait) :

```
<?xml version="1.0" encoding="UTF-8"?>
<osm version="0.6" generator="Overpass API">
<note>The data included in this document is from www.openstreetmap.org. The data is made available under ODbL.</note>
<meta osm_base="2015-12-29T10:24:02Z"/>

  <node id="860901889" lat="44.1377449" lon="4.8093706">
    <tag k="addr:city" v="Orange"/>
    <tag k="addr:country" v="FR"/>
  </node>
  <node id="861724615" lat="44.1352485" lon="4.8121347">
    <tag k="access" v="yes"/>
    <tag k="amenity" v="toilets"/>
  </node>
  <way id="327755238">
    <nd ref="3345693509"/>
    <nd ref="3345693515"/>
    <tag k="amenity" v="parking"/>
    <tag k="source" v="Bing"/>
  </way>
  <way id="374226459">
    <nd ref="3775137769"/>
    <nd ref="3775137762"/>
    <tag k="access" v="yes"/>
    <tag k="amenity" v="parking"/>
  </way>
  <relation id="1654142">
    <member type="way" ref="43671610" role="label"/>
    <member type="way" ref="43671611" role="label"/>
    <tag k="addr:postcode" v="84100"/>
    <tag k="addr:street" v="Boulevard Édouard Daladier"/>
  </relation>
  <relation id="3371895">
    <member type="way" ref="251208243" role="outer"/>
    <member type="way" ref="134904848" role="inner"/>
    <tag k="FR:ERP" v="PA-1"/>
    <tag k="addr:city" v="Orange"/>
</osm>
```

**** Option tags ****

Récupération des identifiants et des attributs des objets (sans géométrie)

La requête suivante :

```
[out:xml][timeout:25];
(
  node["amenity"]({{bbox}});
  way["amenity"]({{bbox}});
  relation["amenity"]({{bbox}});
);
out tags;
```

renvoi les données suivantes (extrait) :

La requête suivante :

```
[out:xml][timeout:25];
(
  node["amenity"]({{bbox}});
  way["amenity"]({{bbox}});
  relation["amenity"]({{bbox}});
);
out tags;
```

Les requêtes pour Overpass Turbo
renvoi les données suivantes (extrait) :

```
<?xml version="1.0" encoding="UTF-8"?>
<osm version="0.6" generator="Overpass API">
<note>The data included in this document is from www.openstreetmap.org. The data is made available under ODbL.</note>
<meta osm_base="2015-12-29T10:35:02Z"/>

 <node id="860901889">
  <tag k="addr:city" v="Orange"/>
  <tag k="addr:country" v="FR"/>
 </node>
 <node id="861724615">
  <tag k="access" v="yes"/>
  <tag k="amenity" v="toilets"/>
 </node>
 <way id="327755238">
  <tag k="amenity" v="parking"/>
  <tag k="source" v="Bing"/>
 </way>
 <way id="374226459">
  <tag k="access" v="yes"/>
  <tag k="amenity" v="parking"/>
 </way>
 <relation id="1654142">
  <tag k="addr:postcode" v="84100"/>
  <tag k="addr:street" v="Boulevard Édouard Daladier"/>
 </relation>
 <relation id="3371895">
  <tag k="FR:ERP" v="PA-1"/>
  <tag k="addr:city" v="Orange"/>

</osm>
```

**** Option meta ****

Récupération des identifiants, de la géométrie, des attributs et des métadonnées des objets

La requête suivante :

```
[out:xml][timeout:25];
(
 node["amenity"]({{bbox}});
 way["amenity"]({{bbox}});
 relation["amenity"]({{bbox}});
);
out meta;
```

renvoi les données suivantes (extrait) :

```
<?xml version="1.0" encoding="UTF-8"?>
<osm version="0.6" generator="Overpass API">
<note>The data included in this document is from www.openstreetmap.org. The data is made available under ODbL.</note>
<meta osm_base="2015-12-29T10:45:02Z"/>

 <node id="860901889" lat="44.1377449" lon="4.8093706" version="5" timestamp="2014-08-11T07:29:29Z" changeset="24671373" uid="425317" user="tony emery">
  <tag k="addr:city" v="Orange"/>
  <tag k="addr:country" v="FR"/>
 </node>
 <node id="861724615" lat="44.1352485" lon="4.8121347" version="6" timestamp="2014-08-05T21:02:09Z" changeset="24564298" uid="188930" user="J-Louis ZIMMERMANN">
  <tag k="access" v="yes"/>
  <tag k="amenity" v="toilets"/>
 </node>
 <way id="327755238" version="1" timestamp="2015-02-12T13:14:47Z" changeset="28795232" uid="188930" user="J-Louis ZIMMERMANN">
  <nd ref="3345693509"/>
  <nd ref="3345693515"/>
  <tag k="amenity" v="parking"/>
  <tag k="source" v="Bing"/>
 </way>
 <way id="374226459" version="1" timestamp="2015-10-07T05:19:02Z" changeset="34483251" uid="188930" user="J-Louis ZIMMERMANN">
  <nd ref="3775137769"/>
  <nd ref="3775137762"/>
  <tag k="access" v="yes"/>
  <tag k="amenity" v="parking"/>
 </way>
 <relation id="1654142" version="7" timestamp="2015-07-09T19:43:54Z" changeset="32528180" uid="242257" user="Emmanuel Dewaele">
  <member type="way" ref="43671610" role="label"/>
  <member type="way" ref="43671611" role="label"/>
  <tag k="addr:postcode" v="84100"/>
  <tag k="addr:street" v="Boulevard Édouard Daladier"/>
```

```
 </relation>
 <relation id="3371895" version="3"
timestamp="2015-08-15T18:42:37Z"
changeset="33357109" uid="2772976"
user="bot_gileri">
  <member type="way" ref="251208243"
role="outer"/>
  <member type="way" ref="134904848"
role="inner"/>
  <tag k="FR:ERP" v="PA-1"/>
  <tag k="addr:city" v="Orange"/>
 </relation>

</osm>
```

**** Options d'emprise****

```
out meta bb;
 <way id="374226459">
  <bounds minlat="44.1377297" minlon="4.8116675" maxlat="44.1377970" maxlon="4.8119160"/>
  <nd ref="3775137769"/>
 <relation id="1654142">
  <bounds minlat="44.1354812" minlon="4.8120821" maxlat="44.1359775" maxlon="4.8127095"/>
  <member type="way" ref="43671610" role="label"/>

out meta center;
 <way id="374226459">
  <center lat="44.1377634" lon="4.8117918"/>
  <nd ref="3775137769"/>
 <relation id="1654142">
  <center lat="44.1357294" lon="4.8123958"/>
  <member type="way" ref="43671610" role="label"/>
```

Requêtes par limites géographiques

Nous allons voir maintenant comment limiter l'emprise géographique des objets que l'on souhaite télécharger.

****Requête QL****

Pour récupérer toutes les données comprises dans une emprise géographiques

```
[out:xml][timeout:180];
(
 node(44.25,4.7,44.30,4.75);
 way(44.25,4.7,44.30,4.75);
 relation(44.25,4.7,44.30,4.75);
);
out meta;
>;
out meta qt;
```

/!\ : ces requêtes peuvent rapidement faire remonter beaucoup d'objets. Les emprises ne doivent pas être trop importantes.
Par exemple, un écart de 0.05° entre 2 coordonnées en x et en y donne 60Mo de données (44.25,4.7,44.30,4.75);

http://overpass-api.de/api/interpreter?data=[out:xml][timeout:180];(node(44.25,4.7,44.30,4.75);way(44.25,4.7,44.30,4.75);relation(44.25,4.7,44.30,4.75););out meta;>;out meta qt;
http://overpass-api.de/api/interpreter?data=%5Bout%3Axml%5D%5Btimeout%3A180%5D%3B%28node%2844%2E25%2C4%2E7%2C44%2E30%2C4%2E75%29%3Bway%2844%2E25%2C4%2E7%2C44%2E30%2C4%2E75%29%3Brelation%2844%2E25%2C4%2E7%2C44%2E30%2C4%2E75%29%3B%29%3Bout%20meta%3B%3E%3Bout%20meta%20qt%3B%0A

Les requêtes pour Overpass Turbo

Requêtes simples

*** Requête QL ***

Pour récupérer les sommets des montagnes dans les Dolomites

```
[out:xml];
area
  [place=region]
  ["region:type"="mountain_area"]
  ["name:en"="Dolomites"];
out body;
node
  [natural=peak]
  (area);
out body qt;
relation
  [place=region]
  ["region:type"="mountain_area"]
  ["name:en"="Dolomites"];
out body;
>;
out skel qt;
```

*** Requête QL ***

Pour récupérer les musées de Vienne

```
[out:xml][timeout:25];
{{geocodeArea:Vienna}}->.SA; (
  node["tourism"="museum"](area.SA);
  way["tourism"="museum"](area.SA);
  relation["tourism"="museum"](area.SA);
);
out body;
>;
out skel qt;
```

Les requêtes pour Overpass Turbo

Exemple avec une emprise géographique {{geocodeArea:Xxx}}

Variable à définir pour récupérer les données d'une commune ou d'une intercommunalité
{{geocodeArea:Caderousse}}
{{geocodeArea:Châteauneuf-du-Pape}}
{{geocodeArea:Courthézon}}
{{geocodeArea:Jonquières, Vaucluse}}
{{geocodeArea:Orange, Vaucluse}}
{{geocodeArea:Communauté de Communes des Pays de Rhône et Ouvèze}}
{{geocodeArea:Vaucluse}}

**** Requête XML ****

Dans une requête XML, il faut ajouter {{geocodeArea:Xxx}} entre *<id-query* et *into="area"/>*

```
<osm-script output="xml" timeout="25">
  <id-query {{nominatimArea:Courthézon}} into="area"/>
  <union>
    <query type="node">
      <has-kv k="highway"/>
      <area-query from="area"/>
    </query>
    <query type="way">
      <has-kv k="highway"/>
      <area-query from="area"/>
    </query>
    <query type="relation">
      <has-kv k="highway"/>
      <area-query from="area"/>
    </query>
  </union>
  <print mode="meta"/>
  <recurse type="down"/>
  <print mode="meta" order="quadtile"/>
</osm-script>
```

**** Requête QL ****

```
[out:xml][timeout:25];
{{geocodeArea:Courthézon}}->.SA;
 (
  node["addr:housenumber"](area.SA);
  way[highway](area.SA);
  relation[type=associatedStreet](area.SA);
  relation["name"="Courthézon"];
);
out meta;
>;
out meta qt;
```

Les requêtes pour Overpass Turbo

Exemple de recherche multiple

Pour récupérer les voies d'une commune à partir de l'ID_INTERNE et les voies qui sont en limite communale

**** Requête QL ****

```
[out:xml][timeout:25];
{{geocodeArea:Caderousse}}->.SA;
(
  way["ref:FR:commune"~"^84027V"];
  relation[highway](area.SA);
  relation["name"="Caderousse"];
);
out meta;
>;
out meta qt;
```

Pour récupérer les voies d'une commune à partir de l'ID_INTERNE à l'échelle de la CCPRO

**** Requête QL ****

```
[out:xml][timeout:25];
(
  way["ref:FR:commune"~"^84016V|^84027V|^84037V|^84039V|^84056V|^84087V|^84129V"];
);
out meta;
>;
out meta qt;
```

Exemple pour faire un plan de ville

Pour récupérer les données de la CCPRO pour faire un plan de ville, on doit lancer plusieurs requêtes en modifiant les variables suivantes.

**** Requête XML ****

Variable à définir

```
  <!-- Ajouter entre <query type= et </query> -->
    <has-kv k="amenity"/>
    <has-kv k="barrier"/>
    <has-kv k="building"/>
    <has-kv k="craft"/>
    <has-kv k="highway"/>
    <has-kv k="housenumber"/>
    <has-kv k="landuse"/>
    <has-kv k="leisure"/>
    <has-kv k="natural"/>
    <has-kv k="office"/>
    <has-kv k="parking"/>
    <has-kv k="railway"/>
    <has-kv k="recycling"/>
    <has-kv k="Waterway"/>
```

```
<osm-script output="xml">
  <print mode="meta"/>
  <query type="node">
    <has-kv k="highway"/>
    <bbox-query s=«43.98» w=«4.70» n=«44.27» e=«4.99»/>
  </query>
  <query type="way">
    <has-kv k="highway"/>
    <bbox-query s=«43.98» w=«4.70» n=«44.27» e=«4.99»/>
```

Les requêtes pour Overpass Turbo

```
  </query>
  <print mode="meta"/>
  <recurse type="down"/>
  <print mode="meta"/>
</osm-script>
```

**** Requête QL ****

Variable à définir
{{geocodeArea:Bédarrides}}
{{geocodeArea:Caderousse}}
{{geocodeArea:Châteauneuf-du-Pape}}
{{geocodeArea:Courthézon}}
{{geocodeArea:Jonquières, Vaucluse}}
{{geocodeArea:Orange, Vaucluse}}
{{geocodeArea:Sorgues}}
{{geocodeArea:Communauté de Communes des Pays de Rhône et Ouvèze}}
{{geocodeArea:Vaucluse}}

```
 node[amenity]
 way[amenity]
 node[barrier]
 way["building"="yes"]
 way["building"!="yes"]
 node[craft]
 way[craft]
 node["addr:housenumber"]
 way[landuse]
 way[leisure]
 way[natural]
 node[office]
 way[office]
 node[parking]
 way[parking]
 way[railway]
 node[recycling]
 node[shop]
 way[shop]
 way[waterway]
```

```
[out:xml][timeout:55];
{{geocodeArea:Communauté de Communes des Pays de Rhône et Ouvèze}}->.SA;
(
  node[landuse](area.SA);
  way[landuse](area.SA);
  relation[landuse](area.SA);
);
out meta;
>;
out meta qt;
```

Les requêtes pour Overpass Turbo

Limites administratives

**** Requête XML ****

Variable à définir

Pour récupérer les contours des communes

<has-kv k="boundary" v="administrative"/>
<has-kv k="admin_level" v="8"/>

Pour récupérer les contours des intercommunalités

<has-kv k="boundary" v="local_authority"/>

Option à ajouter entre *<query type=* et *</query>*

<has-kv k="name" v="Communauté de Communes des Pays de Rhône et Ouvèze"/>
<has-kv k="name" v="Sorgues"/>

```
<osm-script output="xml" timeout="25">
  <union>
    <query type="node">
      <has-kv k="boundary" v="administrative"/>
      <has-kv k="admin_level" v="8"/>
      <has-kv k="name" v="Sorgues"/>
      <bbox-query s=«43.98» w=«4.70» n=«44.27» e=«4.99»/>
    </query>
    <query type="way">
      <has-kv k="boundary" v="administrative"/>
      <has-kv k="admin_level" v="8"/>
      <has-kv k="name" v="Sorgues"/>
      <bbox-query s=«43.98» w=«4.70» n=«44.27» e=«4.99»/>
    </query>
    <query type="relation">
      <has-kv k="boundary" v="administrative"/>
      <has-kv k="admin_level" v="8"/>
      <has-kv k="name" v="Sorgues"/>
      <bbox-query s=«43.98» w=«4.70» n=«44.27» e=«4.99»/>
    </query>
  </union>
  <print mode="meta"/>
  <recurse type="down"/>
  <print mode="meta" order="quadtile"/>
</osm-script>
```

**** Requête QL ****

Variable à définir

{{geocodeArea:Vaucluse}}
{{geocodeArea:Communauté de Communes des Pays de Rhône et Ouvèze}}
{{geocodeArea:Provence-Alpes-Côtes d'Azur}}
{{geocodeArea:Midi-Pyrénées - Languedoc-Roussillon}}
{{geocodeArea:Auvergne - Rhône-Alpes}}

```
[out:xml][timeout:55];
{{geocodeArea:Vaucluse}}->.SA;
(
  relation[boundary=administrative](area.SA);
  relation[boundary=local_authority](area.SA);
);
```

Les requêtes pour Overpass Turbo

```
out meta;
>;
out meta qt;
```

Exemple pour récupérer les lotissements

Les lotissements doivent être qualifiés à partir de 2 clés :

- Landuse = residential
- Place = neighbourhood

Dans les exemples ci-dessous, nous récupérons tous les « landuse » pour corriger les erreurs

**** Requête XML ****

```
<osm-script output="xml" timeout="25">
<query type="way">
  <has-kv k="landuse" v="residential"/>
  <bbox-query s=«43.98» w=«4.70» n=«44.27» e=«4.99»/>
</query>
<union>
  <item/>
  <recurse type="down"/>
</union>
 <print mode="meta"/>
  <recurse type="down"/>
  <print mode="meta" order="quadtile"/>
</osm-script>
```

**** Requête QL ****

```
[out:xml][timeout:25];
{{geocodeArea:Orange, Vaucluse}}->.SA;
(
  way ["landuse"="residential"](area.SA);
  way ["construction"="residential"](area.SA);
  way ["place"="neighbourhood"](area.SA);
);
out meta;
>;
out meta qt;
```

Exemple sur les voies cyclables

**** Requête QL ****

```
[out:xml][timeout:25];
(
  way["bicycle"](43.98,4.70,44.19,4.96);
  way["highway"="cycleway"](43.98,4.70,44.19,4.96);
  way["cycleway"](43.98,4.70,44.19,4.96);
  relation["route"="bicycle"](43.98,4.70,44.19,4.96);
);
out meta asc;
>;
out meta qt;
```

Récupérer l'éclairage public

**** Requête QL ****

```
[out:xml][timeout:25];
{{geocodeArea:Communauté de Communes des Pays de Rhône et Ouvèze}}->.SA;
(
  node[highway=street_lamp](area.SA);
```

Les requêtes pour Overpass Turbo

```
  node[lit](area.SA);
  way[lit](area.SA);
);
out meta;
>;
out meta qt;
```

Récupérer les relations de l'intercommunalité

Les différents types de relations : cf http://wiki.openstreetmap.org/wiki/FR:Relations

type	Commentaire
associatedStreet ccpro	Ensemble des éléments constituant une rue. Cette relation est particulièrement utilisée afin d'associer le numéro des bâtiments à une rue. Voir Numérotation des rues
boundary ccpro	Ensemble des éléments constituant une frontière, avec enclaves / exclaves. Généralement utilisé pour définir les entités administratives d'un territoire.
bridge	Ensemble des éléments constitutif d'un pont
destination sign	signaux de destination avant ou au niveau des intersections
enforcement ccpro	périphériques de régulation du trafic : radars, radars de feux rouges, balances...
multipolygon ccpro	Ensemble de "ways" fermés permettant de définir une surface, particulièrement utile si cette surface comporte des trous. (devront peut-être être renommées, voir l'article). La relation `type=multipolygon` doit ensuite être taggé de la même manière qu'une simple way fermée (`landuse=*`, `leisure=*`, ...).
public_transport	Partie du schéma OSM des transports publics. Principalement utilisé par `public_transport=stop_area`.
restriction ccpro	pour tout type de restriction de manœuvre
route ccpro	Utilisé pour définir une route, c'est-à-dire un ensemble de voies constituant soit une voie numéroté (A 10, ...), soit une ligne de transport en commun, soit ...
route_master	Regroupement de toutes les relations route pour décrire les trajets dans le transport public.
site ccpro	Relation entre des élèments d'un même site comme une école. **Note**: `site=stop_area` a été remplacé par `public_transport=stop_area`
street ccpro	Similaire à associatedStreet, utilisé principalement à Montévidéo pour associer une rue et des numéros de bâtiments.
tunnel	Ensemble d'éléments constitutifs d'un tunnel
waterway ccpro	Ensemble des éléments constitutifs d'un cours d'eau avec la propriété `waterway=*`

**** Requête XML ****

```
<osm-script output="xml" timeout="65">
  <id-query {{geocodeArea:Communauté de Communes des Pays de Rhône et Ouvèze}} into="area_0"/>
  <union>
    <query type="relation">
      <has-kv k="waterway"/>
      <area-query from="area_0"/>
    </query>
```

```
  </union>
  <union>
    <item/>
    <recurse type="down"/>
  </union>
  <print mode="body"/>
</osm-script>
```

**** Requête QL ****

```
[out:xml][timeout:65];
{{geocodeArea:Communauté de Communes des Pays de Rhône et Ouvèze}}->.SA;
(
 relation[type=associatedStreet](area.SA);
);
out meta;
>;
out meta qt;
```

Récupérer les points de recyclage et les poubelles publiques

**** Requête QL ****

```
[out:xml][timeout:25];
{{geocodeArea:Communauté de Communes des Pays de Rhône et Ouvèze}}->.SA;
(
 node[amenity=waste_basket](area.SA);
 node[amenity=waste_disposal](area.SA);
 node[amenity=recycling](area.SA);
 way[amenity=recycling](area.SA);
);
out meta;
>;
out meta qt;
```

Ressources

Overpass API/Overpass QL : Wiki OpenStreetMap, *http://wiki.openstreetmap.org/wiki/Overpass_API/Overpass_QL*
Référence des tags : Wiki OpenStreetMap,
http://wiki.openstreetmap.org/wiki/FR:%C3%89l%C3%A9ments_cartographiques
Référence requêtes : Wiki OpenStreetMap, *http://wiki.openstreetmap.org/wiki/FR:Overpass_turbo*